Configuring OnStep 4.xx for Instein box

Before We Start

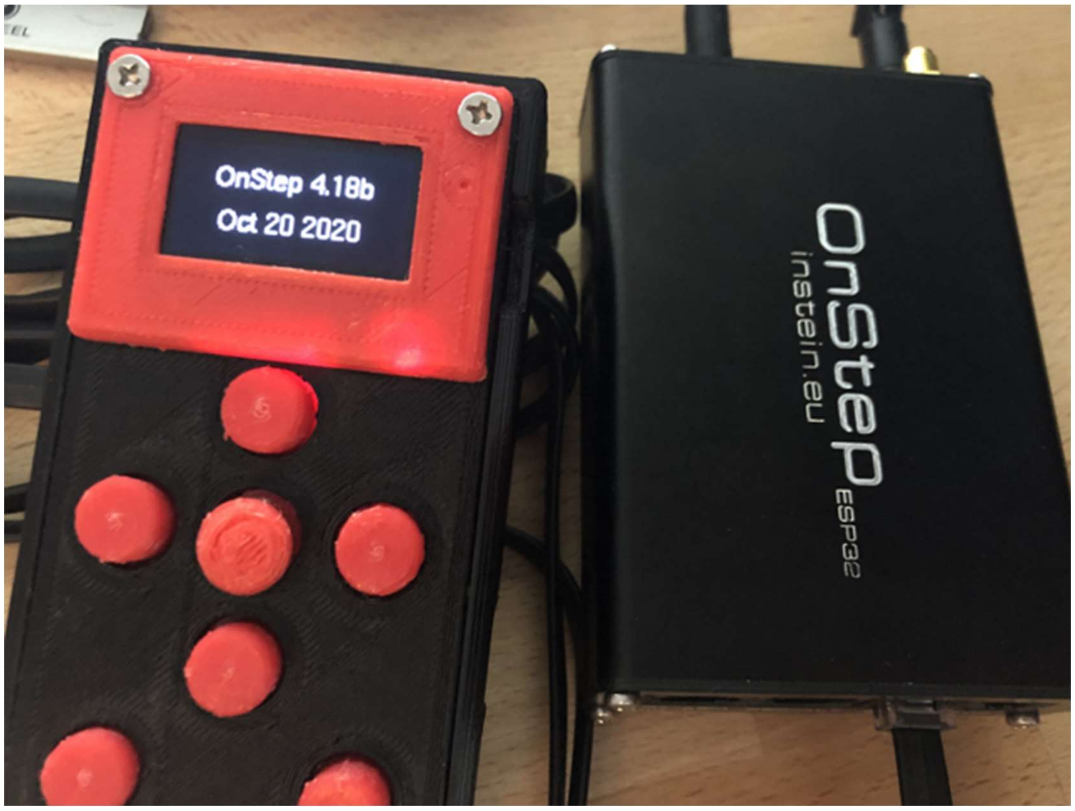
Before using this guide to configure OnStep 4.xx on your Instein box please be advised that I'm in no way associated with Instein or have any knowledge of the product internals besides what is expected from a regular user. As such, the information here is provided "as-is" with not guarantees or commitment from my part. That means, use at your own risk.

What Versions of Instein Does This Document Applies To?

This document and code changes done to OnStep 4.xx branch were the result of analyzing my own Instein box bought around September 2020, and comparing the changes done to OnStep 3.16 code which is the original firmware present in my box.

As there isn't any reliable information on what different hardware versions were available from Instein in the past, I'm describing here what I had inside my box and what hardware changes were described in the original sources code (see above).

This document then applies to any Instein box based on ESP32 main processor. It's not clear when they changed to using ESP32, but the box indicated on the outside that it's based on ESP32, as seen below:



Instein OnStep controller from September 2020 contain the following components inside:

- Espressif ESP32-WROOM-32U
- ESP-07S wifi module
- TMC2130
- Motors Nema17 0.9o 17HM15-0904S (5,4VDC, 0,9A, 36Ncm)

Additionally, the OnStep 3.16 source code shared by Instein had comments indicating that after September 6th, all Instein boxes came with TMC 5160 stepper drivers by default (see here), so if you bought one before this date, you might have the change the config to the proper driver.

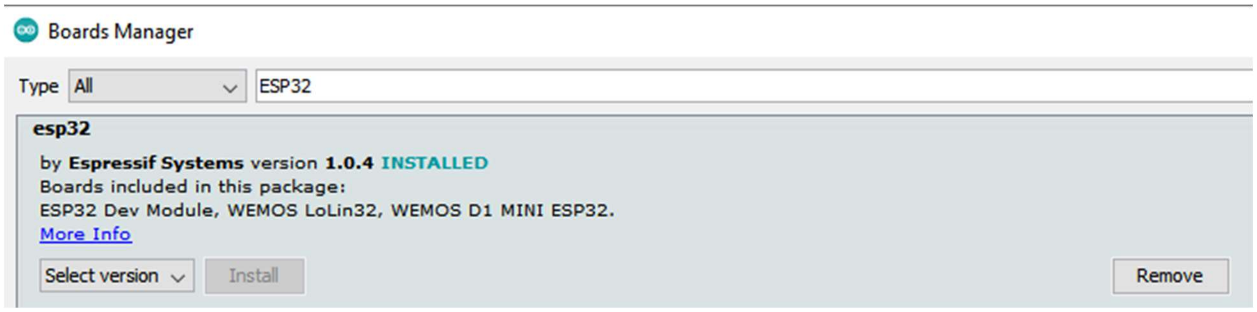
Preparing the Arduino Environment for Instein

In order to compile the OnStep 4.xx branch for Instein, there are a few of libraries that need to be installed in the Arduino environment but most of this is already documented in the OnStep Wiki. Please be sure to read the page below and follow the steps describing how to configure your environment for ESP32:

- https://onstep.groups.io/g/main/wiki/3915

It is important to make sure the versions of the libraries being installed are the same as described in the page. When installing your ESP32 and ESP8266 board management libraries, you need to make sure they look like this:

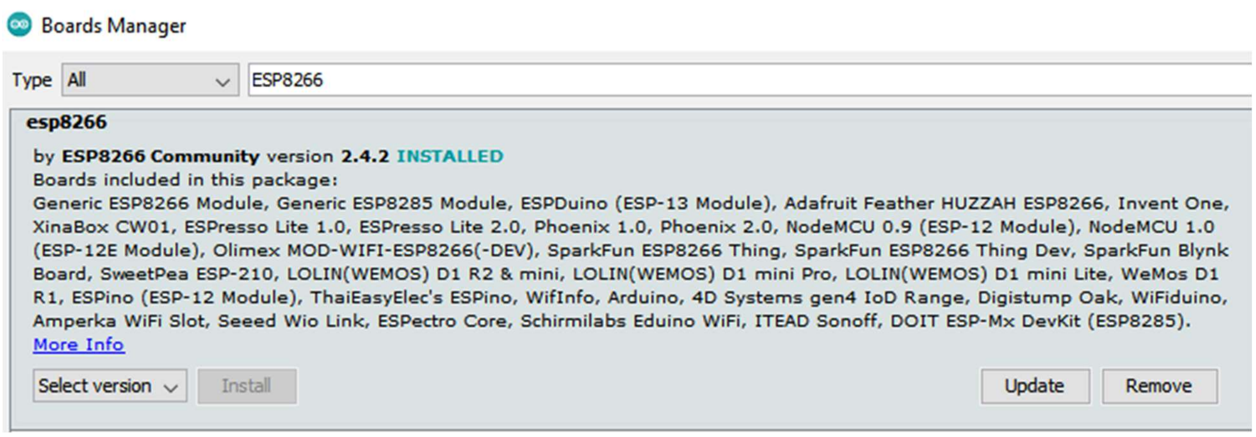
ESP32 Board:



esp32

by **Espressif Systems** version **1.0.4** INSTALLED
Boards included in this package:
ESP32 Dev Module, WEMOS LoLin32, WEMOS D1 MINI ESP32.
More Info

ESP8266 Board:



esp8266

by **ESP8266 Community** version **2.4.2** INSTALLED
Boards included in this package:
Generic ESP8266 Module, Generic ESP8285 Module, ESPDuino (ESP-13 Module), Adafruit Feather HUZZAH ESP8266, Invent One, XinaBox CW01, ESPresso Lite 1.0, ESPresso Lite 2.0, Phoenix 1.0, Phoenix 2.0, NodeMCU 0.9 (ESP-12 Module), NodeMCU 1.0 (ESP-12E Module), Olimex MOD-WIFI-ESP8266(-DEV), SparkFun ESP8266 Thing, SparkFun ESP8266 Thing Dev, SparkFun Blynk Board, SweetPea ESP-210, LOLIN(WEMOS) D1 R2 & mini, LOLIN(WEMOS) D1 mini Pro, LOLIN(WEMOS) D1 mini Lite, WeMos D1 R1, ESPino (ESP-12 Module), ThaiEasyElec's ESPino, WifInfo, Arduino, 4D Systems gen4 IoD Range, Digistump Oak, WiFiduino, Amperka WiFi Slot, Seeed Wio Link, ESPectro Core, Schirmilabs Eduino WiFi, ITEAD Sonoff, DOIT ESP-Mx DevKit (ESP8285).
More Info

Additionally, if you have an original Smart Hand Controller (SHC) from Instein or want to build one based on the projects shared in the OnStep community, you may need to install the following libraries:

- U8G2 by Oliver v2.28.6 Library (required only for SmartHandController)
- Ephemeris by Marscaper (single version library, required only for SmartHandController)

To install these, go to your Arduino IDE->Tools->Manage Libraries, then search for the two libraries above and install them.

After installing the required libraries, you can proceed and download the master branch from OnStep official repository from one of the URLs below:

- https://github.com/hjd1964/OnStep.git (if you're using GIT)
- https://github.com/hjd1964/OnStep/archive/master.zip (if you just want the latest source code)

Please remember to rename the downloaded folder from "OnStep-master" to OnStep before opening it in the Arduido IDE.

Configuring the Firmware

One of the main differences between Instein controller and other OnStep projects is their use of an extra serial device to control the AUX port in the box. This serial device uses some pins in ESP32 which were assigned to other tasks in the original OnStep firmware.

This difference required manual changes to the HardwareSerial.cpp file in the past, but the code has now been integrated into the main firmware code by Howard Dutton, which means the extra serial device can be enabled via config.h by specifying the PINMAP for Instein.

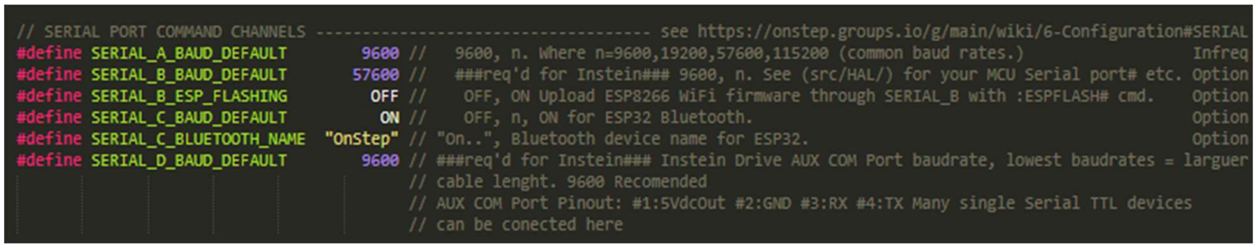
My recommendation for anyone trying to configure OnStep 4.xx on their Instein controller is to **NOT** use the original config file present in the master repository, but instead use the custom example I provide here:

- https://github.com/intel0ut/OnStep/blob/Instein_mods/Config.InsteinESP1.h

Download the file above, and replace the original "config.h" from the 4.xx branch with the version above.

The main differences are related to the serial device configuration, and I recommend these settings are not touched or they may break the communication with the Wifi module, the SHC or any auxiliary device the user may want to connect to AUX port.

These are the default values which are known to work for Instein:



```
// SERIAL PORT COMMAND CHANNELS ------------------------------------ see https://onstep.groups.io/g/main/wiki/6-Configuration#SERIAL
#define SERIAL_A_BAUD_DEFAULT         9600 //   9600, n. Where n=9600,19200,57600,115200 (common baud rates.)                Infreq
#define SERIAL_B_BAUD_DEFAULT        57600 //   ###req'd for Instein### 9600, n. See (src/HAL/) for your MCU Serial port# etc. Option
#define SERIAL_B_ESP_FLASHING          OFF //    OFF, ON Upload ESP8266 WiFi firmware through SERIAL_B with :ESPFLASH# cmd.    Option
#define SERIAL_C_BAUD_DEFAULT           ON //    OFF, n, ON for ESP32 Bluetooth.                                              Option
#define SERIAL_C_BLUETOOTH_NAME  "OnStep" // "On..", Bluetooth device name for ESP32.                                         Option
#define SERIAL_D_BAUD_DEFAULT         9600 // ###req'd for Instein### Instein Drive AUX COM Port baudrate, lowest baudrates = larguer
                                           // cable lenght. 9600 Recomended
                                           // AUX COM Port Pinout: #1:5VdcOut #2:GND #3:RX #4:TX Many single Serial TTL devices
                                           // can be conected here
```

Following the config section for Serial devices, there are several optional devices supported by OnStep which are **NOT** supported by Instein.

As said before, Instein controllers have an AUX port which they mention in their website can be used for external devices like GPS or temperature probes, but since I don't own one, I cannot test how to configure such devices with the box. If anyone is able to connect one of these devices to Instein controller feel free to ping me on the main OnStep list and let me know what to do and I can update the document.

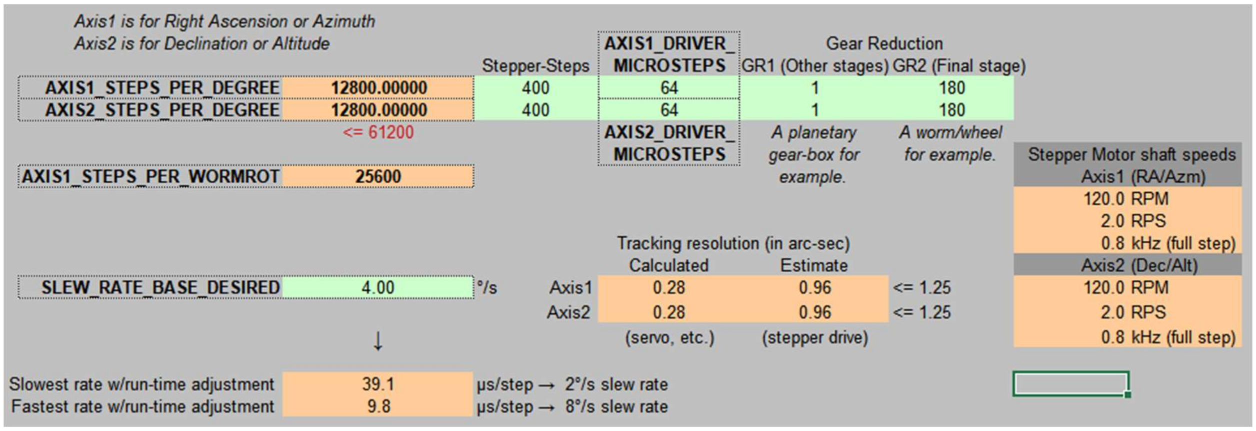
One device that is supported by Instein and you may want to configure here is the ST4 hand controller. You can either use the SHC provided by Instein (at a cost) or build your own like I did. Mine was based on ESP32 (see here), and worked fine with Instein controller without any modification.

The final step in the firmware configuration is to set your Axis. This section requires you to know in advance the characteristics of your mount, as well as what stepper drivers are present in the box you own.

In order to find the values to use in the config, I recommend you use the excellent spreadsheet created by Khalid Baheyeldin here:

- http://www.stellarjourney.com/assets/downloads/OnStep-Calculations.xls

You will need to enter steps per revolution from your stepper motor (Stepper-steps in the XLS), the microsteps you plan to use (usually 64 for TMC 2130 0r TMC 5160) and your gear reduction (GR1 and GR2, which depends on your mount). In my case the values look like below:



Axis1 is for Right Ascension or Azimuth
Axis2 is for Declination or Altitude

		Stepper-Steps	AXIS1_DRIVER_MICROSTEPS	GR1 (Other stages)	GR2 (Final stage)
AXIS1_STEPS_PER_DEGREE	12800.00000	400	64	1	180
AXIS2_STEPS_PER_DEGREE	12800.00000	400	64	1	180
	<= 61200		AXIS2_DRIVER_MICROSTEPS	*A planetary gear-box for example.*	*A worm/wheel for example.*
AXIS1_STEPS_PER_WORMROT	25600				

Gear Reduction

Stepper Motor shaft speeds
Axis1 (RA/Azm)
120.0 RPM
2.0 RPS
0.8 kHz (full step)
Axis2 (Dec/Alt)
120.0 RPM
2.0 RPS
0.8 kHz (full step)

Tracking resolution (in arc-sec)

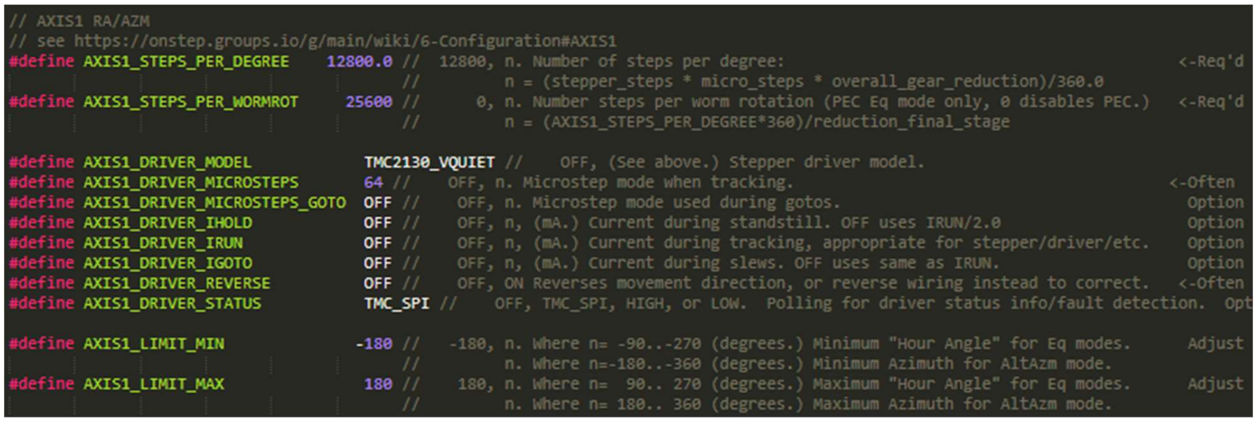
		Calculated	Estimate	
SLEW_RATE_BASE_DESIRED 4.00 °/s	Axis1	0.28	0.96	<= 1.25
	Axis2	0.28	0.96	<= 1.25
		(servo, etc.)	(stepper drive)	

↓

Slowest rate w/run-time adjustment	39.1	μs/step → 2°/s slew rate
Fastest rate w/run-time adjustment	9.8	μs/step → 8°/s slew rate

The SLEW_RATE_BASE_DESIRED value will tell OnStep what is the maximum accepted slew rate, and you need to configure this according to your taste. Usually you can set this to a value like 6, and after uploading the firmware once, see in the web interface what is the maximum slew rate your controller can handle (it's around 5degree/sec in ESP32) then use a SLEW_RATE_BASE_DESIRED a little smaller than that to avoid issues.

The example below is my configuration for a Losmandy GM8 using the parameters above:



```
// AXIS1 RA/AZM
// see https://onstep.groups.io/g/main/wiki/6-Configuration#AXIS1
#define AXIS1_STEPS_PER_DEGREE    12800.0 //  12800, n. Number of steps per degree:                                          <-Req'd
                                          //         n = (stepper_steps * micro_steps * overall_gear_reduction)/360.0
#define AXIS1_STEPS_PER_WORMROT     25600 //      0, n. Number steps per worm rotation (PEC Eq mode only, 0 disables PEC.)   <-Req'd
                                          //         n = (AXIS1_STEPS_PER_DEGREE*360)/reduction_final_stage

#define AXIS1_DRIVER_MODEL            TMC2130_VQUIET //    OFF, (See above.) Stepper driver model.
#define AXIS1_DRIVER_MICROSTEPS       64 //    OFF, n. Microstep mode when tracking.                                        <-Often
#define AXIS1_DRIVER_MICROSTEPS_GOTO  OFF //    OFF, n. Microstep mode used during gotos.                                    Option
#define AXIS1_DRIVER_IHOLD            OFF //    OFF, n, (mA.) Current during standstill. OFF uses IRUN/2.0                   Option
#define AXIS1_DRIVER_IRUN             OFF //    OFF, n, (mA.) Current during tracking, appropriate for stepper/driver/etc.   Option
#define AXIS1_DRIVER_IGOTO            OFF //    OFF, n, (mA.) Current during slews. OFF uses same as IRUN.                   Option
#define AXIS1_DRIVER_REVERSE          OFF //    OFF, ON Reverses movement direction, or reverse wiring instead to correct.  <-Often
#define AXIS1_DRIVER_STATUS           TMC_SPI //    OFF, TMC_SPI, HIGH, or LOW.  Polling for driver status info/fault detection.  Opt

#define AXIS1_LIMIT_MIN              -180 //   -180, n. Where n= -90..-270 (degrees.) Minimum "Hour Angle" for Eq modes.      Adjust
                                          //         n. Where n=-180..-360 (degrees.) Minimum Azimuth for AltAzm mode.
#define AXIS1_LIMIT_MAX               180 //    180, n. Where n=  90.. 270 (degrees.) Maximum "Hour Angle" for Eq modes.      Adjust
                                          //         n. Where n= 180.. 360 (degrees.) Maximum Azimuth for AltAzm mode.
```

As you can see, I have TMC2130_VQUIET in my driver model parameter. You will need to change this to **TMC5160_QUIET** if your model was built after September 6th. You will also need to change the parameters below according to the example:

```
#define AXIS1_DRIVER_IHOLD          400

#define AXIS1_DRIVER_IRUN           600

#define AXIS1_DRIVER_IGOTO          900
```

You then need to do the same for AXIS2. Please note that if you see the motors running in the wrong direction after uploading the firmware, you will need to set AXIS1_DRIVER_REVERSE to ON in the axis showing the issue.

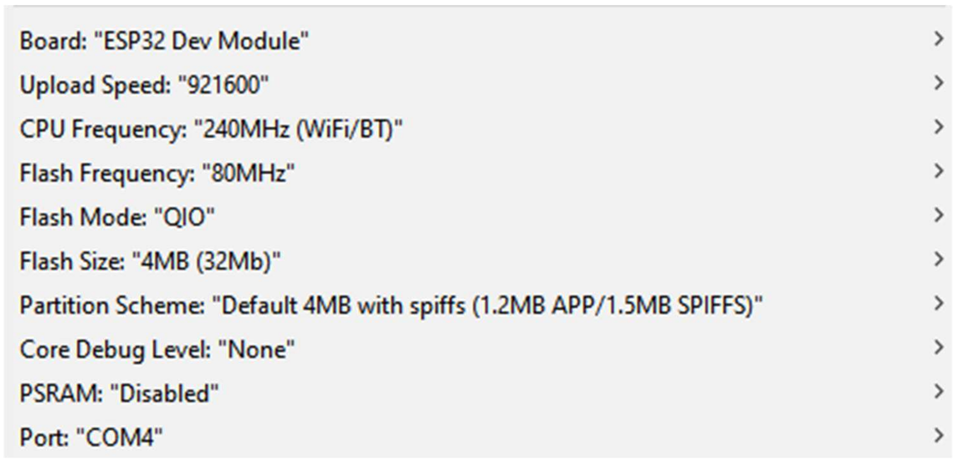
I recommend you keep a separate copy of your config file so if you want to upgrade your device in the future with the master branch, you can use the config from your copy with the correct parameters needed by Instein.

Uploading the firmware

Uploading the firmware to Instein controller requires a USB cable to connect to your computer with the Arduino IDE, and to change the switch in the box to MAIN or WIFI depending on what you're uploading.

Once your configuration is done, change the switch in the box to the MAIN position (usually the side of the Antennas), connect the micro USB cable, connect a 12v power adapter and turn the box on. This should enable a new serial device in your computer (COM port in Windows or /dev/ttyUSBx in Linux).

Open in the Arduino IDE the file OnStep.ino from the master branch you downloaded earlier, and configure the board device as shown below:



Board: "ESP32 Dev Module"
Upload Speed: "921600"
CPU Frequency: "240MHz (WiFi/BT)"
Flash Frequency: "80MHz"
Flash Mode: "QIO"
Flash Size: "4MB (32Mb)"
Partition Scheme: "Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS)"
Core Debug Level: "None"
PSRAM: "Disabled"
Port: "COM4"

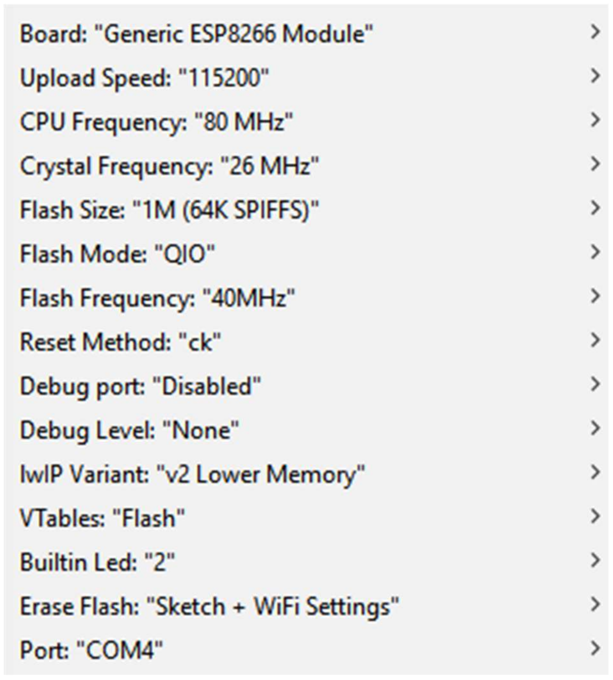
Note port might differ depending on your OS or COM port configuration. The values above should be the default values once you choose the ESP32 Dev Module.

If everything is alright, you should be able to click in Sketch->Upload and the upload will happen after the firmware compilation is done.

Turn the box off, remove the USB cable, then change the switch to the WIFI position. It's important to remove the USB cable before changing the switch or Windows might not recognize the device again.

Close the Arduino IDE with OnStep and open the file addons/WIFI/WIFI.ino to upload the wifi module firmware.

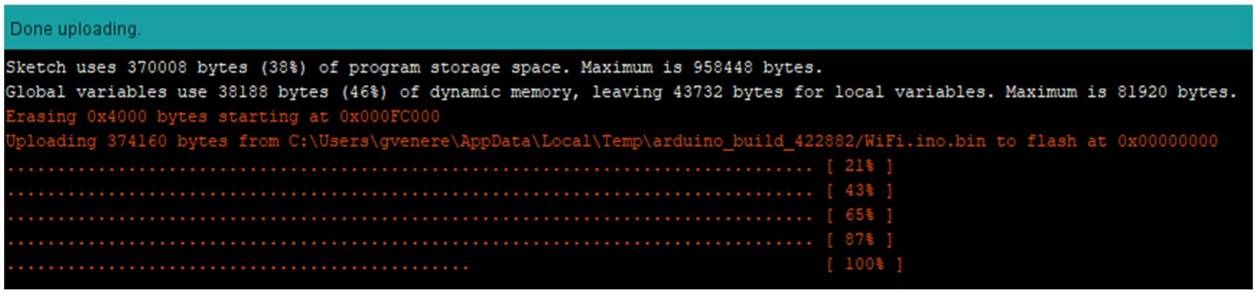
This time, you need to change the board to ESP8266 and change a few options as shown below:



Board: "Generic ESP8266 Module"
Upload Speed: "115200"
CPU Frequency: "80 MHz"
Crystal Frequency: "26 MHz"
Flash Size: "1M (64K SPIFFS)"
Flash Mode: "QIO"
Flash Frequency: "40MHz"
Reset Method: "ck"
Debug port: "Disabled"
Debug Level: "None"
lwIP Variant: "v2 Lower Memory"
VTables: "Flash"
Builtin Led: "2"
Erase Flash: "Sketch + WiFi Settings"
Port: "COM4"

Note the "Flash size" option was changed to "1M (64K SPIFFS)" and the "Erase Flash" option was changed to "Sketch + WiFi Settings".

Click upload Sketch again, and after some time you should see this message indicating the flash process is done:



```
Done uploading.
Sketch uses 370008 bytes (38%) of program storage space. Maximum is 958448 bytes.
Global variables use 38188 bytes (46%) of dynamic memory, leaving 43732 bytes for local variables. Maximum is 81920 bytes.
Erasing 0x4000 bytes starting at 0x000FC000
Uploading 374160 bytes from C:\Users\gvenere\AppData\Local\Temp\arduino_build_422882/WiFi.ino.bin to flash at 0x00000000
................................................................................ [ 21% ]
................................................................................ [ 43% ]
................................................................................ [ 65% ]
................................................................................ [ 87% ]
.............................................                                    [ 100% ]
```

The process is now done. Power off the unit, disconnect the USB cable, and change the switch back to the middle position so it can operate normally.

Follow the steps in the main OnStep Wiki to use the WiFi module and test the device with your motors to confirm everything is working.